# Perturbation Mappings in Polynomiography

Krzysztof Gdawiec

Institute of Computer Science, University of Silesia
Będzińska 39, 41-200, Sosnowiec, Poland
kgdawiec@ux2.math.us.edu.pl

**Abstract.** In the paper, a modification of rendering algorithm of polynomiograph is presented. Polynomiography is a method of visualization of complex polynomial root finding process and it has applications among other things in aesthetic pattern generation. The proposed modification is based on a perturbation mapping, which is added in the iteration process of the root finding method. The use of the perturbation mapping alters the shape of the polynomiograph, obtaining in this way new and diverse patterns. The results from the paper can further enrich the functionality of the existing polynomiography software.

**Keywords:** polynomiography, perturbation, aesthetic pattern, computer art

## 1 Introduction

Today, one of the aims in computer aided design is to develop methods that make the artistic design and pattern generation much easier. Usually the most work during a design stage is carried out by a designer manually. Especially, in the cases in which the graphic design should contain some unique unrepeatable artistic features. Therefore, it is highly useful to develop an automatic method for aesthetic patterns generation. In the literature we can find many different methods, e.g., method based on Iterated Function Systems [13], method for creating stone-like decorations using marbling [11]. A very interesting method is polynomiography [7]. It is based on the root finding methods of polynomials with complex coefficients.

In this paper we present a modification of the standard rendering algorithm used in polynomiography. The modification is based on the use of perturbation mapping before the use of root finding method in the standard algorithm. The perturbation mapping disturbs the process of finding the roots of polynomial thereby obtaining new and diverse patterns comparing to the standard polynomiography.

The paper is organized as follows. In Sec. 2 we introduce some basic information about polynomiography and a standard algorithm for rendering polynomiographs. Then, in Sec. 3 we present perturbation mapping and its use in the polynomiography for obtaining new patterns. Some examples of polynomiographs obtained with the proposed modifications are presented in Sec. 4. Finally, in Sec. 5 we give some concluding remarks.

## 2 Polynomiography

The notion of polynomiography appeared in the literature about 2000 and was introduced by Kalantari. Polynomiography is defined as the art and science of visualization in approximation of the zeros of complex polynomials, via fractal and non-fractal images created using the mathematical convergence properties of iteration functions [6]. Single image created using the mentioned methods is called polynomiograph.

In polynomiography the main element is the root finding method. Many different root finding methods exist in the literature, e.g., Newton method [6], Traub-Ostrowski method [1], Harmonic Mean Newton's method [1], Steffensen method [10], and also we can find families of root finding method, e.g., Basic Family [6], Parametric Basic Family [6], Euler-Schröder Family [6], Jarratt Family [2]. Let us recall two root finding methods, that will be used in the examples presented in Sec. 4.

Let us consider a polynomial $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ of the form:

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0. \tag{1}$$

The Newton root finding method is given by the following formula:

$$N(z) = z - \frac{p(z)}{p'(z)}, \tag{2}$$

and Halley root finding method by the formula:

$$H(z) = z - \frac{2p'(z)p(z)}{2p'(z)^2 - p''(z)p(z)}. \tag{3}$$

To render a single polynomiograph we can use Algorithm 1. It is a basic rendering algorithm. In the literature we can find other methods of rendering polynomiographs, which are based on the ideas taken from the Mandelbrot and Julia set rendering algorithms [4]. Moreover, we can replace the Picard iteration used in the algorithm with other iteration methods [5], e.g., Mann, Ishikawa, Noor. In the algorithm we use the so-called iteration colouring, i.e., colour is determined according to the number of iteration in which we have left the while loop. Other colouring methods exist in the literature, e.g., basins of attraction, mixed colouring [6].

## 3 Perturbation Mappings in Polynomiography

In Algorithm 1 for any $z_0$ we can treat the sequence $\{z_0, z_1, z_2, \ldots\}$ as the orbit of $z_0$. For different starting points $z_0$ using the same root finding method we obtain different orbits. So, if we change some point in the orbit of a given starting point, then the orbit changes starting from the altered point. In this way we can obtain alternation of the polynomiographs shape.

**Algorithm 1:** Rendering of polynomiograph

**Input**: $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – polynomial, $A \subset \mathbb{C}$ – area, $M$ – number of iterations, $\varepsilon$ – accuracy, $R : \mathbb{C} \to \mathbb{C}$ – root finding method, $colours[0..k]$ – colourmap.

**Output**: Polynomiograph for the area $A$.

```
1 for z_0 ∈ A do
2     i = 0
3     while i ≤ M do
4         z_{i+1} = R(z_i)
5         if |z_{i+1} − z_i| < ε then
6             break
7         i = i + 1
8     Print z_0 with colours[i] colour
```

Let us modify line 4 in Algorithm 1 in a following way:

$$z_{i+1} = (R \circ \rho)(z_i, i+1) = R(\rho(z_i, i+1)), \tag{4}$$

where $\rho : \mathbb{C} \times \mathbb{N} \to \mathbb{C}$ is a mapping. Moreover, we modify the convergence test in line 5 in a following way:

$$|z_{i+1} - \rho(z_i, i+1)| < \varepsilon. \tag{5}$$

The mapping $\rho$ is called perturbation mapping and its aim is to alter (perturb) the orbit during the iteration process. Because we can alter the orbit in very different ways, so we do not make any assumptions about the perturbation mapping. Let us notice that when $\rho(z, i) = z$ for all $z \in \mathbb{C}$ and $i \in \mathbb{N}$, then (4) and (5) reduce to the standard iteration and convergence test used in the polynomiography.

The simplest perturbation mapping that alters the orbit is addition of a fixed complex number $v$, i.e.,

$$\rho_v(z, i) = z + v. \tag{6}$$

The value of $v$ cannot be arbitrary, because we will lose the convergence of the root finding method and the resulting polynomiograph will be a rectangle filled with one colour. From the conducted research it turns out that the value $v$ is highly dependent on $\varepsilon$. The modulus of $v$ can be greater than $\varepsilon$ only by a small value. Taking into account this observation it is very comfortable to represent $v$ in the trigonometric form:

$$v = r\varepsilon(\cos\theta + \mathbf{i}\sin\theta), \tag{7}$$

where $r \in [0, 1.1]$ and $\theta \in [0, 2\pi)$.

Another example of perturbation mapping is mapping that uses different values of $v$ in subsequent iterations, e.g.,

$$\rho_m(z,i) = \begin{cases} z + v_1, & \text{if } i \mod m = 0, \\ z + v_2, & \text{if } i \mod m = 1, \\ \dots & \\ z + v_m, & \text{if } i \mod m = m - 1, \end{cases} \tag{8}$$

where $m \in \mathbb{N}$ and $v_1, v_2, \dots, v_m \in \mathbb{C}$.

The examples of perturbation mappings presented so far are all deterministic. It is tempting to use randomness to obtain random patterns. But it turns out that the polynomiographs generated using random value of $v$ in each iteration does not give a random pattern. We obtain a very similar noisy patterns, so their appearance is not aesthetic. Instead of using pure randomness we can use the random number generator of computer graphics [8], i.e., a noise function.

Besides the use of perturbation mapping we can also take combination of the standard iteration and the perturbed one. Let $\rho$ be a given perturbation mapping and $R$ a root finding method. We define new iteration process in the following way:

$$z_{i+1} = \alpha R(z_i) + (1 - \alpha) R(\rho(z_i, i + 1)), \tag{9}$$

where $\alpha \in \mathbb{C}$. Let us notice that for $\alpha = 1$ iteration (9) reduces to the standard iteration used in the polynomiography, and for $\alpha = 0$ it reduces to (4). So the combined iteration process is more general than the iteration with perturbation mapping.

Algorithm 2 presents method for rendering polynomiograph using the combined iteration process.

**Algorithm 2:** Rendering of polynomiograph with combined iteration

**Input**: $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – polynomial, $A \subset \mathbb{C}$ – area, $M$ – number of iterations, $\varepsilon$ – accuracy, $R : \mathbb{C} \to \mathbb{C}$ – root finding method, $\rho : \mathbb{C} \times \mathbb{N} \to \mathbb{C}$ – perturbation mapping, $\alpha \in \mathbb{C}$ – parameter, $colours[0..k]$ – colourmap.

**Output**: Polynomiograph for the area $A$.

```
for z_0 ∈ A do
    i = 0
    while i ≤ M do
        w = ρ(z_i, i + 1)
        z_{i+1} = αR(z_i) + (1 − α)R(w)
        if |z_{i+1} − w| < ε then
            break
        i = i + 1
    Print z_0 with colours[i] colour
```

## 4 Examples

In this section, we present some examples of polynomiographs obtained using the proposed modifications from Sec. 3. To visually compare the obtained patterns with the originals ones we start by presenting the patterns obtained with the standard rendering algorithm (Algorithm 1). The patterns are presented in Fig. 1, and the parameters used to generate them were the following:

(a) $p(z) = z^3 - 1$, $A = [-1.5, 1.5]^2$, $M = 15$, $\varepsilon = 0.001$, Newton's root finding method,
(b) $p(z) = z^5 + z$, $A = [-2.0, 2.0]^2$, $M = 15$, $\varepsilon = 0.001$, Halley's root finding method,
(c) $p(z) = z^3 - 3z + 3$, $A = [-2.5, 2.5]^2$, $M = 20$, $\varepsilon = 0.001$, Newton's root finding method.

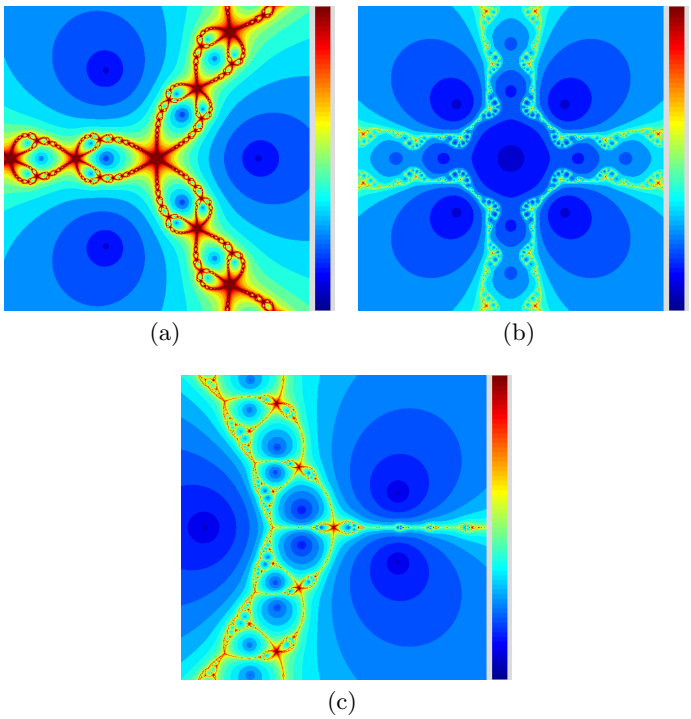


**Fig. 1.** Polynomiographs generated with the standard rendering algorithm.

The first example presents the use of perturbation mapping with the addition of a fixed complex number (6). The parameters to generate the polynomiographs were the same as in Fig. 1(a), and the complex numbers used in the perturbation mapping had modulus equal to $\varepsilon$ and their arguments were the following: (a)

$\theta = 0.00$, (b) $\theta = 0.22$, (c) $\theta = 0.50$, (d) $\theta = 0.99$. The obtained polynomiographs are presented in Fig. 2.

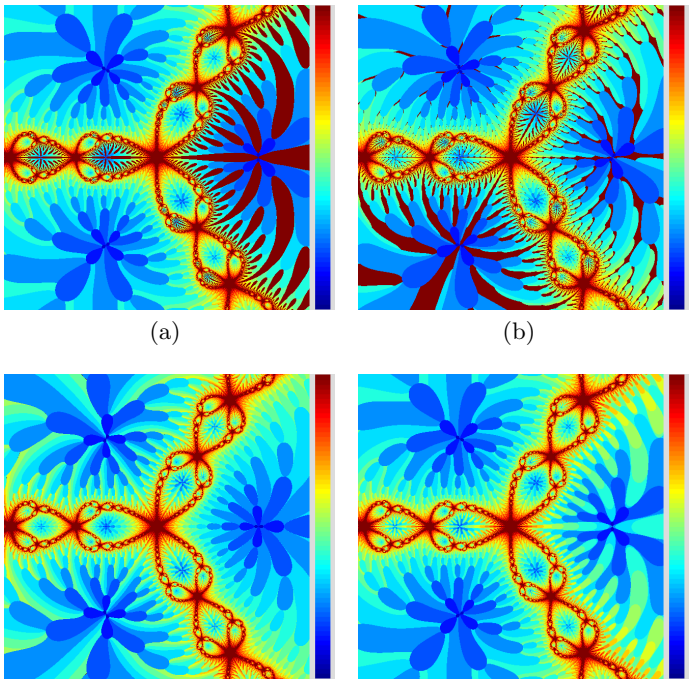


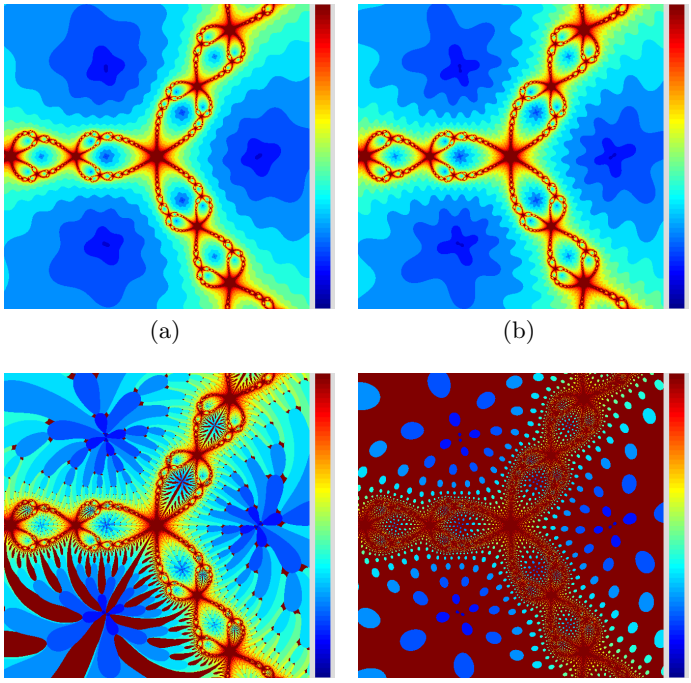

**Fig. 2.** Polynomiographs obtained using the perturbation mapping (6) – different arguments.

The second example presents the influence of the modulus of the fixed complex number used in the perturbation mapping on the polynomiograph. The parameters to generate the polynomiographs were the same as in Fig. 1(a), and the complex numbers used in the perturbation mapping had argument equal to $0.6\pi$ and their moduli were the following: (a) $0.6\varepsilon$, (b) $0.9\varepsilon$, (c) $1.0\varepsilon$, (d) $1.1\varepsilon$. The obtained polynomiographs are presented in Fig. 3.

The next example presents the use of perturbation mapping given by (8). The parameters to generate the polynomiographs were the same as in Fig. 1(b), and the parameters of the complex numbers used in the perturbation mapping were the following:

(a)

$$\begin{cases} r = 0.95, \theta = 0.6\pi, & \text{if } i \mod 2 = 0, \\ r = 1.1, \theta = 1.5\pi, & \text{if } i \mod 2 = 1, \end{cases} \tag{10}$$



(a) (b)

(c) (d)

**Fig. 3.** Polynomiographs obtained using the perturbation mapping (6) – different moduli.

(b)

$$\begin{cases} r = 1.1, \theta = 1.5\pi, & \text{if } i \mod 2 = 0, \\ r = 0.95, \theta = 0.6\pi, & \text{if } i \mod 2 = 1, \end{cases} \tag{11}$$

(c)

$$\begin{cases} r = 1.1, \theta = 1.62\pi, & \text{if } i \mod 3 = 0, \\ r = 0.5, \theta = 0.98\pi, & \text{if } i \mod 3 = 1, \\ r = 1.0, \theta = 0.20\pi, & \text{if } i \mod 3 = 2, \end{cases} \tag{12}$$

(d)

$$\begin{cases} r = 1.0, \theta = 1.62\pi, & \text{if } i \mod 3 = 0, \\ r = 1.0, \theta = 0.49\pi, & \text{if } i \mod 3 = 1, \\ r = 1.01, \theta = 0.2\pi, & \text{if } i \mod 3 = 2. \end{cases} \tag{13}$$

The obtained polynomiographs are presented in Fig. 4.

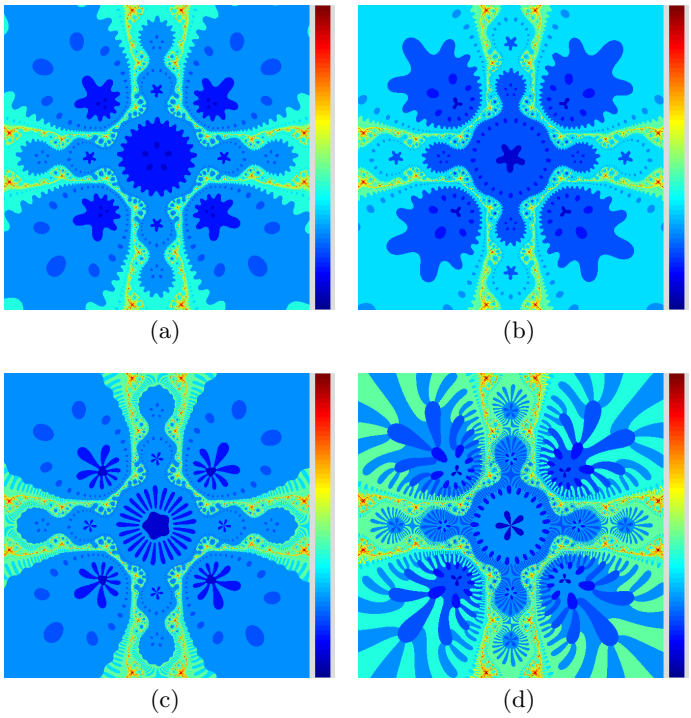


**Fig. 4.** Polynomiographs obtained using the perturbation mapping (8).

The last example presents the use of combined iteration process (9). The parameters to generate the polynomiographs were the same as in Fig. 1(c), the

perturbation mapping was given by (6) with $v = \varepsilon(\cos\pi + \mathbf{i}\sin\pi)$, and the values of $\alpha$ were the following: (a) $-300 + 10\mathbf{i}$, (b) $-10$, (c) 100, (d) 200. The obtained polynomiographs are presented in Fig. 5.

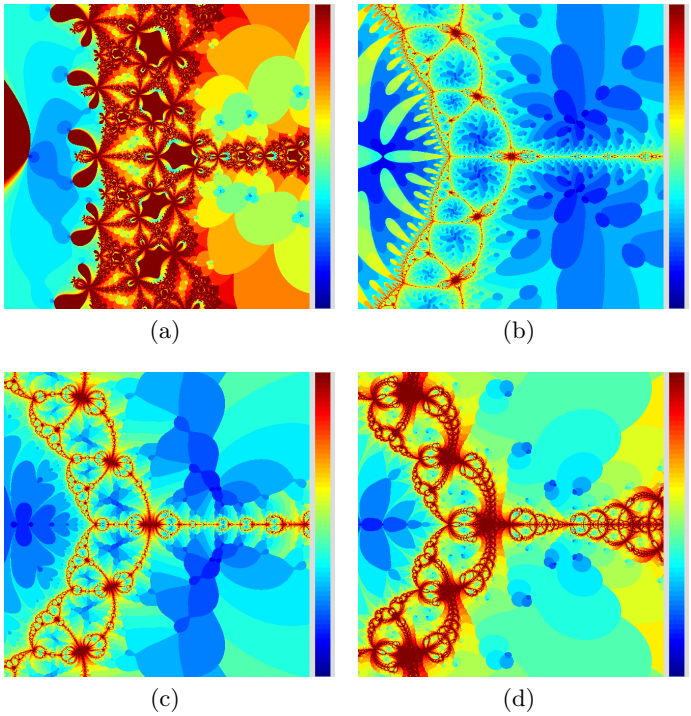



**Fig. 5.** Polynomiographs obtained using the combined iteration process (9).

## 5 Conclusions

In this paper, we presented a modification of the standard rendering algorithm for polynomiographs. The modification was based on the use of a perturbation mapping. The mapping was added in the iteration process of the root finding method in two ways. In the first method we used the perturbation mapping before the root finding method, and in the second method we used combination of the original root finding method and its perturbed version. Moreover, the convergence test of the algorithm was modified. The presented examples show that using the proposed methods we are able to obtain very interesting and diverse patterns, that differ from the original patterns obtained with the standard polynomiography.

In our further work we will try to extend the results of the paper by using the q-system numbers [9] and bicomplex numbers [12] instead of the complex

numbers. Moreover, we will try to bring the perturbation mappings into the quaternion Newton method [3] and to develop an algorithm of visualization of the quaternionic root finding process in 3D.

## References


1. Ardelean, G.: A Comparison Between Iterative Methods by Using the Basins of Attraction. Applied Mathematics and Computation 218(1), 88-95 (2011)
2. Chun, C., Neta, B., Kim, S.: On Jarratt's Family of Optimal Fourth-Order Iterative Methods and Their Dynamics. Fractals 22(4), 1450013 (2014)
3. Falcão, M.I.: Newton Method in the Context of Quaternion Analysis. Applied Mathematics and Computation 236, 458-470 (2014)
4. Gdawiec, K.: Mandelbrot- and Julia-like Rendering of Polynomiographs. In: Chmielewski, L.J., et al. (eds.) ICCVG 2014. LNCS, vol. 8671, pp. 25-32. Springer International Publishing (2014)
5. Gdawiec, K., Kotarski, W., Lisowska, A.: Polynomiography Based on the Nonstandard Newton-like Root Finding Methods. Abstract and Applied Analysis, vol. 2015, Article ID 797594, 19 pages (2015)
6. Kalantari, B.: Polynomial Root-Finding and Polynomiography. World Scientific, Singapore (2009)
7. Kalantari, B.: Two and Three-dimensional Art Inspired by Polynomiography. In: Proceedings of Bridges, Banff, Canada, pp. 321-328 (2005)
8. Lagae, A., Lefebvre, S., Cook, R., De Rose, T., Drettakis, G., Ebert, D., Lewis, J., Perlin, K., Zwicker, M.: State of the Art in Procedural Noise Functions. In: Hauser, H., Reinhard, E. (eds.) EG'10: Proceedings of State of the Art Reports, Norrköping, Sweden, pp. 1-19 (2010)
9. Levin, M.: Discontinuous and Alternate Q-System Fractals. Computer & Graphics 18(6), 873-884 (1994)
10. Liu, X.-D., Zhang, J.-H., Li, Z.-J., Zhang, J.-X.: Generalized Secant Methods and Their Fractal Patterns. Fractals 17(2), 211-215 (2009)
11. Lu, S., Jaffer, A., Jin, X., Zhao, H., Mao, X.: Mathematical Marbling. IEEE Computer Graphics and Applications 32(6), 26-35 (2012)
12. Wang, X.-Y., Song, W.-J.: The Generalized M-J Sets for Bicomplex Numbers. Nonlinear Dynamics 72(1-2), 17-26 (2013)
13. Wannarumon, S., Bohez, E.L.J., Annanon, K.: Aesthetic Evolutionary Algorithm for Fractal-based User-centered Jewelry Design. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22(1), 19-39 (2008)